Unsupervised Learning for Music Genre Classification of Song Lyrics

Manav Aggarwal, Varin Nair, Tiger Sun

CA: Richard Diehl Martinez

CS 221

Fall 2018

1 Overview

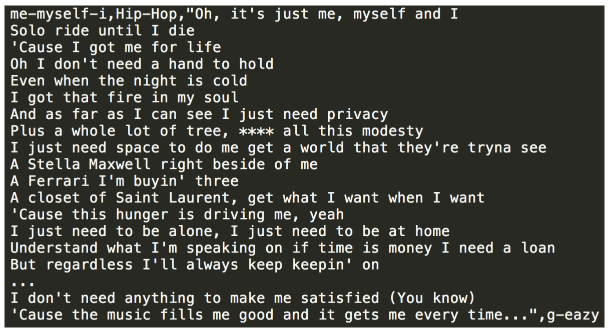
1.1 Introduction and Motivations

Our project aims to try to classify song genres into clusters based on lyrics with a sufficiently high accuracy. We will explore various models to vectorize our lyrics and cluster those lyrics into our genres. Current methods for clustering lyrics be genre is usually done through unsupervised learning, such as using regression or neural networks. We decided to explore solving this problem through an unsupervised learning approach instead.

It will be interesting to compare how lyrically similar songs of completely different genres are to each other. For example, if we ran a Kanye West song through our program, it would be quite surprising if it were actually also very lyrically close to the country music genre. A potential application of this is seeing how the audio and nonverbal cues can entirely the change the categorization of a song despite having similar lyrics to a completely different genre. Additionally, we would love the ability to input any text file, such as a writing sample or political speech, and classify it into a genre.

1.2 Data Filtering and Infrastructure

Even though we will be using unsupervised learning, we still initially used labeled data to train and test our model's validity. We first found a Kaggle dataset associating lyrics to its correct genre. The Kaggle dataset had a mix of many genres, such as pop, hip-hop, country, rock, metal, folk, jazz. However, for the purposes of our algorithm, we decided to filter the data and only use hip-hop, country, rock, and metal lyrics. These genres and pop had the highest lyric counts which would give us a better model. However, the genre of pop encompasses many of the other genres, which is why we decided to filter it out during our clustering step, which we will talk about more below. It could be interesting afterwards to see how pop songs would really be classified, if pop were not a valid genre.



```
me-myself-i,Hip-Hop,"Oh, it's just me, myself and I
Solo ride until I die
'Cause I got me for life
Oh I don't need a hand to hold
Even when the night is cold
I got that fire in my soul
And as far as I can see I just need privacy
Plus a whole lot of tree, **** all this modesty
I just need space to do me get a world that they're tryna see
A Stella Maxwell right beside of me
A Ferrari I'm buyin' three
A closet of Saint Laurent, get what I want when I want
'Cause this hunger is driving me, yeah
I just need to be alone, I just need to be at home
Understand what I'm speaking on if time is money I need a loan
But regardless I'll always keep keepin' on
...
I don't need anything to make me satisfied (You know)
'Cause the music fills me good and it gets me every time...",g-eazy
```

Figure 1: Excerpt of lyric data

To obtain more data, we also used Genius API to scrape lyrics from our desired genres. We compiled lists of 40 artists from each genre, and used a library to query Genius lyrics.

In total, we compiled around 71365 songs, which we used to train our model. We used 57092 in our clustering algorithm.

1.3 Literature Review

Previous attempts have used supervised learning methods to classify lyrics by genre. For example, Tsaptsinos trains hierarchical attention networks, which tries to identify segments of lyrics with the most important to a genre, to create a model. [5] Similarly, other projects have included neural networks, such as LSTM and GRU, which are other examples of supervised learning. [6] In their paper, Fell and Sporleder create a feature vectors based on various characteristics of the lyrics, such as length and top 100 genre specific n-grams, to predict the genre of the song. [2]

Based on the literature, it seemed like that no one had tried using unsupervised learning methods to classify lyrics. Labeled data for supervised learning can be hard to obtain, and we wanted to see if using an unsupervised method would also grant similar results.

1.4 General Approach

After filtering and cleaning our data, we would like to input it into some model that would turn each lyric into a vector. We would then run a clustering algorithm to create our genres. Afterwards, depending on either testing or recreational use, we would feed in a lyric or text and see which cluster or mixture of clusters that text would be associated with. This could help us determine the accuracy of our model. We go into more detail regarding our approach in the next couple pages.

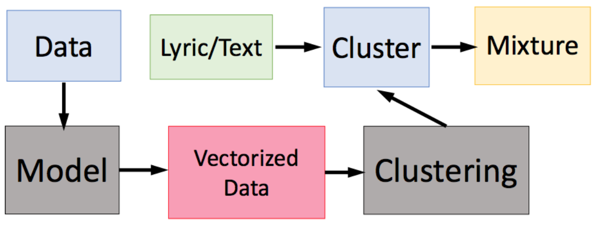


Figure 2: Flow chart of procedure

2 Baseline and Oracle

For our baseline, we implemented the bag-of-words method as our model. In this model, the vector associated with each lyric are the counts of each word in the lyric. For example, for text: "the cat and the angry cat," the vector would be $\{the : 2,\ cat : 2,\ and : 1,\ angry : 1\}$. Afterwards, we clustered the vectors using a k-means algorithm with cosine distance. There are a few obvious issues with the system, such as the order of words is rendered insignificant where in reality, the order of the words could be very important. We chose cosine distance so that the length of the lyric (number of words in the lyric) would have less of an effect on the vector. Running the baseline with $k = 3$ and $k = 4$ clusters, we have the following results.

Cluster Number	Hip-hop	Rock	Metal	Country
0	5633	187	202	79
1	2901	9726	11840	10278
2	5739	4360	2231	3916

Table 1: Cluster distributions for baseline with k = 3

Cluster Number	Hip-hop	Rock	Metal	Country
0	4371	126	129	45
1	3839	948	1031	758
2	2781	9090	11337	9515
3	3282	4109	1776	3955

Table 2: Cluster distributions for baseline with k = 4

3 Language Processing Models

The first step to clustering is to create vector representations for each lyric. This way, we can perform vector operations in the process of clustering.

3.1 Global Vectors for Word Representation

Global Vectors for Word Representation (GloVe) is a technique used to transfer specific words to vectors. In essence, it creates a weighted co-occurrence matrix for each word based on proximity to nearby words, and uses an algorithm to reduce the dimensionality of that matrix. This is useful for preserving semantic meaning of a word. For example, consider the vectors for 'Queen', 'King,' 'Man', and 'Woman.' Then, 'King' - 'Man' + 'Woman' = 'Queen' since meaning is encoded into the vectors. The ordering and structure of the phrase affects the weights of the words within the phrase, which is something that bag-of-words could not accomplish.

In our case, we used the Stanford GloVe pre-processed data from Common Crawl Wikipedia that contained 840 billion tokens, 2.2 million vocabulary words with case-preserved mapped to 300-d vectors. This time, we included pop, rock, hip-hop, metal, and country as genres. For each song lyric, we used the sum of the word vectors assigned to each word in that lyric. Then, we ran a k-means clustering algorithm on the lyrics of the songs, using the cosine distance function, similar to our bag-of-words idea.

Running k-means with $k = 5$, we achieved these clusters:

Cluster Number	Hip-hop	Rock	Metal	Country	Pop
0	3427	4359	3015	4520	5173
1	1337	1785	4543	1638	1429
2	3950	5611	4556	5928	4672
3	4582	1629	1328	1618	1652
4	977	888	831	567	1347

Table 3: Cluster distributions for GloVe with k = 5

We originally included pop music in our clustering algorithm for GloVe, which gave us 5 preliminary clusters. However, the results were not promising, as each cluster seems to have equal anounts of all songs, so we decided to go down another path, described below.

3.2 Paragraph Vectors

After working with the general bag of words model consisting of frequency counts of the words in lyrics and GloVe, we decided to try a more sophisticated model. We realized that these word-to-vector methods generally lacked the ability to convey the context of a word, which is something we thought would be important.

We decided to use Doc2Vec, which is essentially paragraph embedding that retains word ordering. We considered the lyrics of each song as a document and used the model to create a feature vector. This feature vector includes shared word vectors but also a paragraph vector embedding unique to each document that is able to preserve the relationship between and context of the words. [3]

For our model, feature vectors were all 200-dimensional, and the Doc2Vec model was run for 20 epochs with the distributed memory model to preserve word ordering. We will go more into details regarding our results from this algorithm below.

4 Different Clustering Algorithms

We explored several different clustering algorithms such as basic k-means clustering, Gaussian mixture models (GMMs), and finally density-based spatial clustering of applications with noise (DBSCAN). To visualize our higher dimension data in just two dimensions, we will use principal component analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE).

4.1 Data Visualization Technique

Though data visualization ultimately will not affect our results, it can help give a better idea of how exactly our data points are being clustered, and can give a more cursory glance of the effectiveness of our techniques.

4.1.1 PCA

PCA is a mathematical algorithm used to reduce the dimension of vectors using eigenvectors and eigenvalues. In essence, it seeks to use linear algebra to find the greatest amount of variation using the fewest number of variables.

4.1.2 t-SNE

Instead of using hard mathematics, t-SNE uses a more probabilistic approach that is outside the scope of this research project. It is essentially aiming to match the original distribution with that of a lower dimension. One of its main drawbacks is that it can be very slow for very large vectors and large amounts of input. To combat this, we will use PCA to minimize each 200-vector down to a 50-vector, and then run t-SNE on these new vectors.[1]

4.2 K-Means

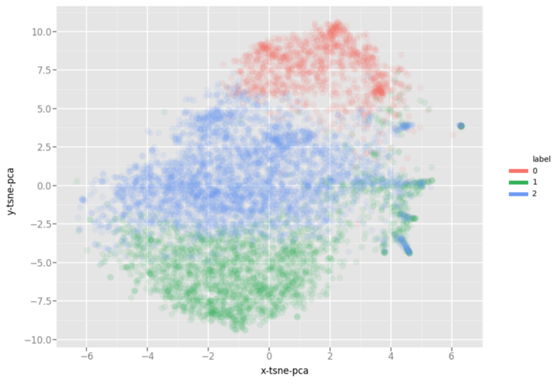
K-means was our baseline technique for clustering. This uses hard clustering where each lyric vector is assigned to the cluster purely depending on the closer centroid. Here are our resulting clusters for $k = 3$ and $k =$ as well as visualizations.

Cluster Number	Hip-hop	Rock	Metal	Country	Label
0	9375	131	249	31	Hip-hop
1	1747	4165	11764	1035	Metal
2	3151	9977	2260	13207	Rock/Country

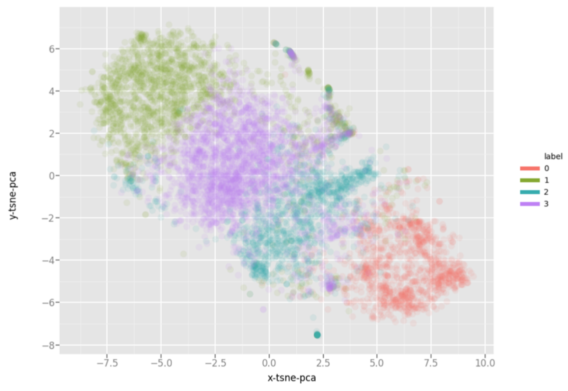
Table 4: Cluster distributions for K-means with k = 3

Cluster Number	Hip-hop	Rock	Metal	Country	Label
0	9139	100	233	12	Hip-hop
1	1473	2951	10502	712	Metal
2	1336	3207	501	5888	Country/Rock
3	2325	8015	3037	7661	Rock/Country

Table 5: Cluster distributions for K-means with k = 4



(a) k = 3 visualization



(b) k = 4 visualization

Figure 3: GMM visualizations - labels correspond to cluster numbers (red: 0, green: 1, blue: 2, purple: 3)

4.3 Hyperparameters

Earlier we mentioned that we used 20 epochs in our training model. We chose this number based on the reconstruction loss of clustering given by our next clustering method, Gaussian Mixture Models. We have attached a graph showing the loss for each epoch in the appendix.

4.4 Gaussian Mixture Models

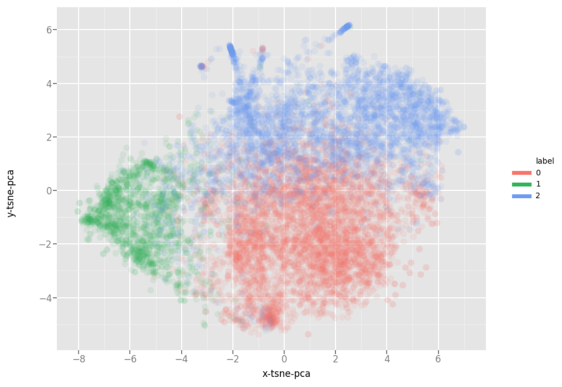
GMMs use a more probabilistic approach than k-means. It uses expectation-maximization (EM) to probabilistically group a lyric into a cluster. When considering a lyric vector, it determines the probability that that vector could be a part of every cluster using a Gaussian probability density function, and then based on those weighted probabilities, randomly assigns that vector to a cluster. This idea of having a mixture is important for us, as we are concerned with the similarity of genres to each other. Thus, information regarding the likelihood of a lyric being assigned into a cluster is important for us. Here are our resulting clusters for $k = 3$ and $k = 4$:

Cluster Number	Hip-hop	Rock	Metal	Country	Label
0	8183	36	174	6	Hip-hop
1	3066	5090	11907	1218	Metal
2	3024	9147	2192	13049	Rock/Country

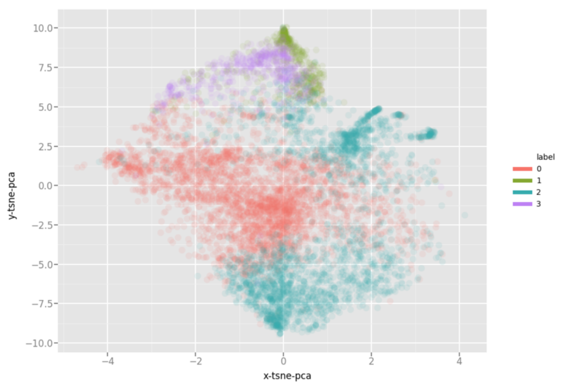
Table 6: Cluster distributions for GMM with k = 3

Cluster Number	Hip-hop	Rock	Metal	Country	Label
0	7858	35	168	6	Hip-hop
1	524	1111	149	3001	Country
2	3552	5587	11618	1889	Metal
3	2339	7540	2338	9377	Country/Rock

Table 7: Cluster distributions for GMM with k = 4



(a) k = 3 visualization



(b) k = 4 visualization

Figure 4: GMM visualizations - labels correspond to cluster numbers (red: 0, green: 1, blue: 2, purple: 3)

5 Next Steps

5.1 DBSCAN

Finally, we considered DBSCAN, which is a density based clustering algorithm. This is especially useful for being able to find clusters of arbitrary shape. Given parameters include ϵ, which is the distance for a neighborhood of point p and n, which is the minimum number of points to classify a region as a neighborhood. Points are classified as core, border, or outlier points. Core points are those with higher density. Border points have low density but are included in the neighborhood of a core point, and finally, outlier points are not in the neighborhood of any core point. DBSCAN is thus especially resistant to outside noise, since it can help factor that into the ϵ distance for each cluster. However, the parameters for ϵ and n were difficult to hone in, and we had trouble finding viable values that would work with our computing constraints and output reasonable values.

5.2 Additional Genres

Next, we could try to incorporate additional genres into our model. We now know that our algorithm works reasonably for our chosen four genres - potentially, future work could increase the number of genres.

6 Success Evaluation and Error Analysis

After performing our clustering algorithm, we marked each cluster using the majority genre (or if there were co-dominant genres, that cluster was considered both of those genres), and measured the accuracy of our genres and clusters. For example, if an actual hip-hop song was in our labeled hip-hop genre, it was correct. For clusters that were codominated, such as by both rock and country, we said that both rock and country were valid and correct labels in measuring accuracy. Additionally, since the clusters from our baseline were difficult to label in that there was no dominant genre, there was no way to measure accuracy. Hence, they have been omitted in this measurement due to poor labels.

Here is a table of our results for each of the clustering algorithms.

k = 3	Accuracy
K-Means	0.7763
GMM	0.6528

Table 8: Accuracy for k = 3

k = 4	Accuracy
K-Means	0.7217
GMM	0.7434

Table 9: Accuracy for k = 4

Additionally, it was suggested to us that we should graph the accuracy of a clustering algorithm with a varying k (number of clusters). Though this might not make sense with our given genres, perhaps it could symbol different subgenres within each genre. We decided accuracy based on the most dominant genre within each cluster. It makes sense that accuracy would increase since there are more groups of similar songs to compare to. For example, if k equaled the number of songs, each songs would be placed in its own cluster and we would have 100% accuracy.

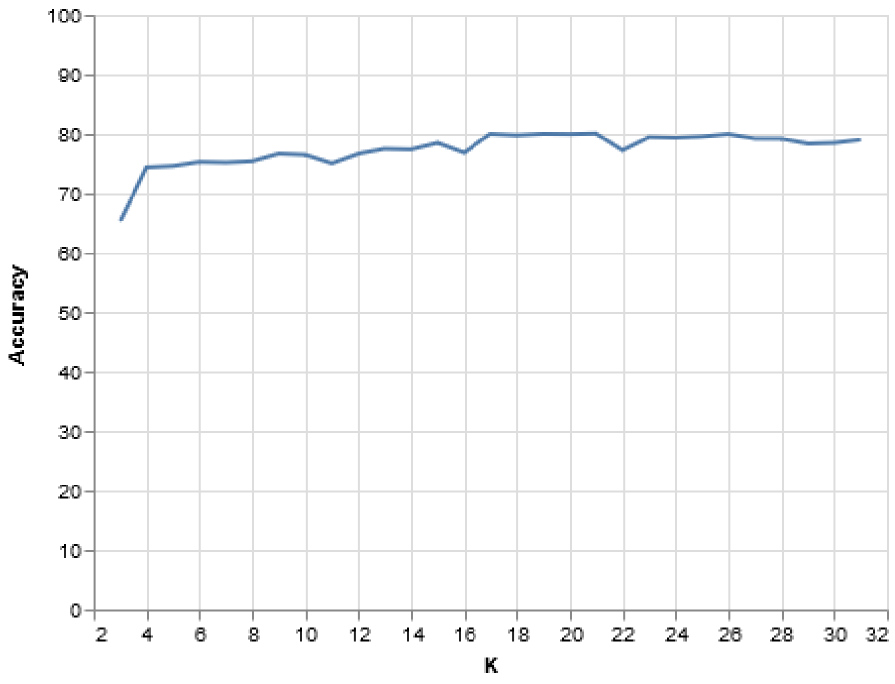


Figure 5: Accuracy vs. K

7 Conclusions

7.1 Analysis

It was much easier to label the clusters for GMM and k-means than the baseline, which shows that these algorithms are indeed effective. Our accuracies were very similar for the k-means and GMM for each of k=3 and k=4. This is expected since we don't expect our clusters to be drastically different in themselves. The reason we choose GMM is to explore the mixtures themselves, since we presume that songs can be mixtures of different genres rather than being hard assigned to a single genre such as in k-means. The accuracies for all were also much higher than the accuracy one would obtain by guessing the genre, which comes out to 33% if we assume rock and country are equivalent since they codominate clusters. Additionally, the fact that rock and country are consistently lumped togethers implies there is great similarity between rock and country lyrics and semantics.

7.2 Extensions

Another interesting application we wanted to explore was to determine which mixture of musical genres any text file would be included in. To show an application of this, we used Martin Luther King Jr.'s "I have a dream speech" with the Gaussian Mixture Model cluster. Let v denote the vector given by this model. In the first table, we have our normalized clusters (percentages of each song within a cluster). In the second table, we have the probabilities that v will be assigned to each cluster. We see that the probability that v will be assigned to cluster two is very high. To determine the mixture of v, we multiply the probability weights by each of the normalized values in the clusters and sum for each category.

Cluster	Hip-hop	Rock	Metal	Country
0	0.974	0.004	0.021	0.001
1	0.112	0.335	0.087	0.465
2	0.145	0.230	0.569	0.051

Table 10: Normalized clusters for GMM with k=3

$p(z_v = 0)$	0.00
$p(z_v = 1)$	0.003
$p(z_v = 2)$	0.997

Table 11: Weights

Our final results for Martin Luther King's speech showed that it was 23.12% Rock, 14.95% Hip-Hop, 5.55% Country, and 56.37% Metal. Running President Trump's inauguration speech through the same process, we achieved similar results: 23.19% Rock, 14.93% Hip- Hop 5.84% Country, 56.04% Metal. We hypothesized that this was because metal music uses conversational language, which is similar to that in a speech.

From our results, we think that unsupervised learning is certainly a viable algorithm for determining genres of music based on just their lyrics, and using probabilistic methods of clustering can allow us to determine mixtures of additional inputs, which could be an interesting idea in the future.

Appendix

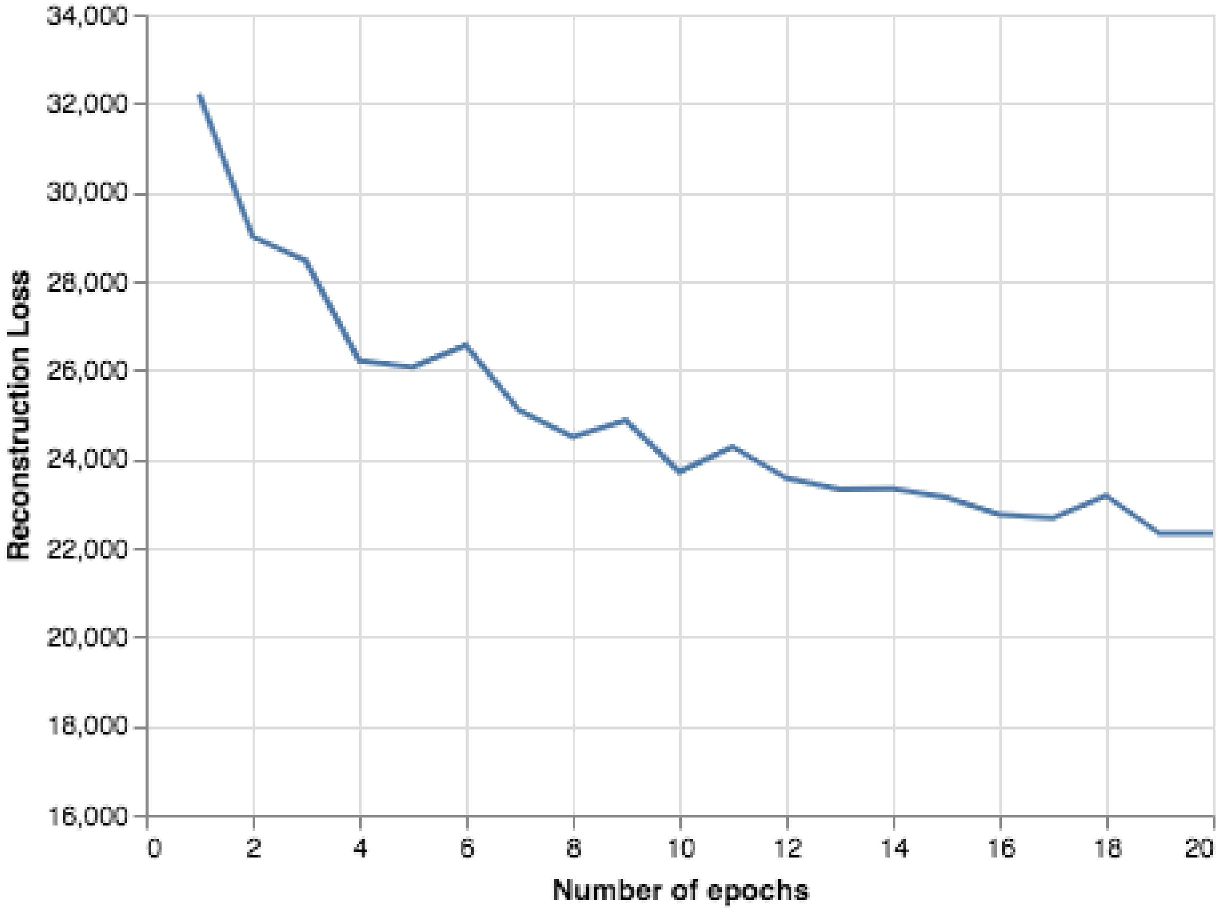


Figure 6: Reconstruction loss v. epochs

References

[1] Derksen, Luuk. *Visualising high-dimensional datasets using PCA and t-SNE in Python* https://medium.com/@luckylwk/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b

[2] Fell, Michael and Sporleder, Caroline. *Lyrics-based Analysis and Classification of Music* http://www.aclweb.org/anthology/C14-1059

[3] Le, Quoc and Mikolov, Tomas. *Distributed Representations of Sentences and Documents* https://arxiv.org/pdf/1405.4053.pdf

[4] Reynolds, Douglas. *Gaussian Mixture Models* https://pdfs.semanticscholar.org/734b/07b53c23f74a3b004d7fe341ae4fce462fc6.pdf

[5] Tsaptsinos, Alexandros. *Music Genre Classification by Lyrics using a Hierarchical Attention Network* https://web.stanford.edu/class/cs224n/reports/2728368.pdf

[6] Zhang, Zhao, and Lu. *Automatic Lyrics-Based Music Genre Classification* https://web.stanford.edu/class/cs224n/reports/6839297.pdf